Assignment 4

```
<?php
```

Question 1 - Check if a string is palindrome or not

```
function checkP($s){
   return strrev($s) == $s ? "true" : "false";
}
echo checkP("madam");   //true
```

Question 2 - Print Table of 73 until 12 times using Recursion

```
function table($i) {
   if($i == 0)
      return;
   table($i - 1);
   echo 73 * $i." ";
}
table(12);  //73 146 219 292 365 438 511 584 657 730 803 876
```

Question 3 - Factorial of a number using Recursion

```
function facto($n) {
   if($n == 0)
      return 1;
   return $n * facto($n - 1);
}
echo facto(5);  //120
```

Question 4 - Sum of first n fibonacci numbers

```
function sumFibo($n) {
   if($n <= 1)
      return $n;
   return sumFibo($n - 1) + sumFibo($n - 2);
}
$n = 5; $s = 0;
for($i = 0; $i < $n;)
   $s += sumFibo($i++);
echo $s;  //7
```

Question 5 - Divide two integers without using *, / or %

```
function div($a, $b) {
   $si = (($a < 0) ^ ($b < 0)) ? -1 : 1;
   $a = abs($a); $b = abs($b); $q = 0;
   for(;$a >= $b; $q++)
```

```
    $a -= $b;
  echo $q * $si;  //-2
}
div(10, -3);
```

Question 6 - Print the count the count of all contiguous substrings that start & end at the same character

```
function subC($s) {
  $count = 0; $n = strlen($s);
  for($i = 0; $i < $n; $i++)
    for($j = 1; $j <= $n - $i; $j++)
      if(equals(substr($s, $i, $j)))
        $count++;
  echo $count;   //5
}
function equals($s) {
  return $s[0] == $s[strlen($s) - 1];
}
subC("good");
```

Question 7 - OOPS

```
class Product {
  public $name, $description, $price;
  function __construct($name, $description, $price) {
    $this->name = $name;
    $this->description = $description;
    $this->price = $price;
  }
  function setName($name) {
    $this->name = $name;
  }
}
$p1 = new Product("Huawei", "Smart phone", 120000);
echo $p1->name."<br>";  //Huawei
$p1->setName("iPhone 12");
echo $p1->name."<br>";  //iPhone 12
$p2 = new Product("Nokia", "Smart phone", 98000);
echo $p2->name."<br>";  //Nokia
$p2->setName("Samsung F12");
echo $p2->name."<br>";  //Samsung F12
```

Question 8 - Find area of a rectangle using inheritance

```
abstract class Shape {
```

```
    protected $l, $b;
    function __construct($l, $b) {
        $this->l = $l;
        $this->b = $b;
    }
    abstract function getArea();
}
class Rectangle extends Shape {
    function __construct($l, $b) {
        parent::__construct($l, $b);
    }
    function getArea() {
        echo "Area: ".$this->l * $this->b;  //Area: 12
    }
}
$r = new Rectangle(2, 6);
$r->getArea();
```

Question 9 - OOPS

```
class BankAccount {
    protected $name, $balance;
    function __construct($name, $balance) {
        $this->name = $name;
        $this->balance = $balance;
        echo "Successfully created account for $name<br>";
        $this->getAcInfo();
    }
    function deposit($amount) {
        if ($amount > 0) {
            $this->balance += $amount;
            echo "Successfully deposited $amount. New balance: $this->balance<br>";
        } else
        echo "Amount should be +ve <br>";
    }
    function withdraw($amount, $at=0) {
        if ($amount <= $this->balance and $amount > 0) {
            $this->balance -= $amount;
            if($at)
                echo "Successfully withdrawn $at. New Balance: $this->balance <br>";
            else
                echo "Successfully withdrawn $amount. New Balance: $this->balance <br>";
        } else
```

```
            echo "Insufficient Balance or Amount should be +ve <br>";
        }
        function getAcInfo() {
            echo "<h3>Account Information</h3>
                Account Holder: $this->name <br>
                Balance: $this->balance<br>";
        }
    }
    class SavingsAc extends BankAccount {
        private $intRate;
        function __construct($name, $balance, $rate) {
            parent::__construct($name, $balance);
            $this->intRate = $rate;
        }
        function calcInt() {
            return $this->balance * $this->intRate / 100;
        }
        function addInterest() {
            $this->balance += $this->calcInt();
            echo "Interest added. New balance: $this->balance";
        }
    }
    class CheckingBalance extends BankAccount {
        protected $count = 0, $num;
        function __construct($name, $bal, $num) {
            parent::__construct($name, $bal);
            $this->num = $num;
        }
        function withdraw($amount, $at=0) {
            if($this->count++ < $this->num) {
                parent::withdraw($amount);
                if($this->count == $this->num)
                    echo "Your next withdrawals will include fees<br>";
            } else {
                $fee = 6;
                parent::withdraw($amount + $fee, $amount);
                echo "This transaction includes fees<br>";
            }
        }
    }
    $p1 = new SavingsAc("Sudipto", 5000, 15.6);
    $p1->deposit(1000);
```

```
$p1->withdraw(1000);
$p1->addInterest();
$p1->getAcInfo();

$p2 = new CheckingBalance("Jacob", 7000, 2);
$p2->deposit(1000);
$p2->withdraw(10);
$p2->withdraw(10);
$p2->withdraw(10);  //Includes withdrawal fees
$p2->withdraw(10);  //Includes withdrawal fees
```

Queation 10 - OOPS

```
class Circle {
    public $r, $pi = 3.14;
    function __construct($r) {
        $this->r = $r;
    }
    function getArea() {
        return "Area: ".($this->pi * $this->r ** 2)."<br>";
    }
    function getPerimater() {
        return "Perimeter: ".(2 * $this->pi * $this->r)."<br>";
    }
}
$c = new Circle(11);
echo  $c->getArea();          //Area: 379.94
echo  $c->getPerimater();     //Perimeter: 69.08
$c = new Circle(4.44);
echo  $c->getArea();          //Area: 61.900704
echo  $c->getPerimater();     //Perimeter: 27.8832
?>
```